Simulación de datos del experimento ATLAS en el superordenador MareNostrum4

Díaz Machado, Elvis

11 de febrero del 2020

Resumen– El experimento del LHC (Gran Colisionador de Hadrones, en Ginebra), ATLAS, genera una cantidad enorme de datos que se han de contrastar con simulaciones, y en este documento se describirán los cambios que se han hecho a las maquinas y componentes responsables de la cola de trabajos, como han afectado a la ejecución de las simulaciones del experimento en el MareNostrum4, y el proceso de ponerlo en producción.

Palabras clave– arc-ce ATLAS BSC contenedores HPC IFAE LHC MareNostrum4 PIC Singularity

Resum– L'experiment de l'LHC (Gran Col·lisionador d'Hadrons, a Ginebra), ATLAS, genera una quantitat enorme de dades que s'han de contrastar amb simulacions, i en aquest document, es descriuran els canvis que s'han fet a les màquines i components responsables de la cua de treballs, com han afectat l'execució de les simulacions de l'experiment en el MareNostrum4, i el procés de posar-lo en producció.

Paraules clau– arc-ce ATLAS BSC contenidors HPC IFAE LHC MareNostrum4 PIC Singularity

Abstract– The LHC's (Large Hadron Collider, in Geneva) experiment, ATLAS, generates a enormous quantity of data that has to be contrasted with simulations, and in this document, we will describe the changes that have been made to the machines and components responsible for the work queue, how they have affected the execution of the simulations of the experiment in the MareNostrum4, and the process of putting it into production.

Keywords– arc-ce ATLAS BSC containers HPC IFAE LHC MareNostrum4 PIC Singularity

1 INTRODUCTION

El experimento ATLAS del LHC genera una enorme cantidad de datos que han de ser contrastados con simulaciones y este trabajo tiene el objeto de procesarlos estos en el superordenador MareNostrum4.

Es necesario mejorar el procesado de datos para poder tratar con la creciente cantidad de datos que generados por el experimento. En un superordenador lo importante es optimizar el código, validarlo y si es posible ejecutarlo de una forma lo más paralela posible. También es importante gestionar los datos de la forma más eficiente posible porque el MareNostrum4 en si no es un repositorio de datos, sólo los transforma.

Desde 2018 en el PIC se ha estado desarrollando un servidor específico, con la misión de enviar y procesar datos en el MareNostum4, que está en funcionamiento y, será la base para el trabajo de desarrollo y optimización.

• Curs 2019/20
• E-mail de contacte: elvis.diaz@e-campus.uab.cat
• Menció realitzada: Enginyeria de Computadors (EC)
• Treball tutoritzat per: Andreu Pérez Martínez (DACSO)

1.1. Objetivos

1. Crear una instancia de desarrollo de un servicio `arc-ce` en el Port d'Informació Científica (PIC) y validarla usando como base una instancia en producción.

2. Registrar la instancia de desarrollo en la base de datos de la colaboración ATLAS para recibir los encargos de la simulación.

3. Poner en servicio la cadena de producción. Asegurarse que todos los cambios están bajo puppet y que los trabajos se ejecutan en el MareNostrum4 y quedan registrados.

4. Actualizar el software instalado en el MareNostrum4 (pasar a CentOS7 las imágenes de Singularity e instalar nuevas versiones del software de simulación).

   4b. Intentar ejecutar alguna imagen *single release* de ATLAS. [5]

5. Optimizar la ejecución de los trabajos en el MareNostrum4: incrementar número de *cores* por trabajo, incrementar número de nodos por trabajo incorporando MPI. Mejorar así el rendimiento de proceso de ejecución del experimento ATLAS en HPC y ponerlo en producción.

   5b. Eventualmente compilar el código con las opciones recomendadas para el MareNostrum4, sin necesidad de contenedores.

1.2. Estado del Arte

El uso de superordenadores por parte del experimento ATLAS se realiza desde hace 5 años. Pero a diferencia del la red *grid*, los superordenadores son muy diferentes entre ellos. Esto requiere una adaptación y optimización diferente en cada caso. El PIC empezó a utilizar el MareNostrum4 en 2018 y muchos experimentos están muy interesados en los métodos de integrar este recurso con el sistema de gestión de trabajos y datos del experimento. En el caso que nos incumbe, ATLAS ha sido pionero. Es estratégico en el PIC usar el superordenador MareNostrum4.

En diciembre 2019 se firmó un acuerdo para que estas actividades de los experimentos del LHC sean consideradas estratégicas para el BSC, con lo que se prevé ampliar la actividad en el MareNostrum4 y siguientes hasta un 13 % de la capacidad total.

1.3. Contexto

1.3.1. LHC

Los experimentos del colisionador de hadrones (LHC) que está situado en Ginebra necesitan grandes cantidades de cálculo para simular las colisiones que se observan experimentalmente. Tradicionalmente se ha usado un *grid* de computación que se conoce como la *World LHC Computing Grid* (WLCG) en el que participan centenares de centros de cálculo de todo el mundo. No obstante, la creciente demanda de computación ha visto en los centros de supercomputación (High Performance Computing Facilities - HPCs) una fuente de recursos al alcance de los investigadores.

1.3.2. ATLAS

En este proyecto, el experimento que recoge los datos de las colisiones del LHC se llama ATLAS (cern.ch/atlas) y el HPC es el MareNostrum4 del BSC en Barcelona. Las instrucciones para definir los parámetros de la simulación se registran en una base de datos en el CERN (Ginebra) y el plan es obtener la información de los datos a simular de esta base de datos, copiar los datos y programas al MareNostrum4, lanzar los trabajos en el sistema de colas, recoger los datos y ponerlos disponibles para los científicos de la colaboración.

La prueba de integración del HPC de Lusitania y MareNostrum4 en el sistema de producción ATLAS comenzó en abril de 2018 en colaboración conjunta entre los centros IFIC y PIC. Desde entonces, se han recibido horas para explotar los HPC españoles. En 2019, a ambos sitios se les concedieron 4 millones de horas en las instalaciones de MaresNostrum (MN4). [1, 17]

2 ARC MIDDLEWARE

La computación en el *grid* tiene tres grandes áreas: computación, almacenamiento e información. El lado del servidor de ARC proporciona servicios para las tres áreas principales: [2] (1.2)

- Elemento de Almacenamiento (SE): el servidor ARC GridFTP además de ser una parte importante del Elemento de Cómputo ARC, también se puede instalar como una solución de almacenamiento independiente.

- Servicio de indexación (EGIIS): *ARC Enhanced Grid Information Indexing Service* es capaz de recopilar registros de recursos de cómputo y elementos de almacenamiento equipados con ARIS (ARC Resource Information Service) y proporcionar estos punteros de recursos a las herramientas del cliente. Hay varias instancias de EGIIS implementadas en todo el mundo. Los nuevos recursos generalmente se registran en uno o más de los índices existentes.

- Elemento de Cómputo (CE): mediante la instalación del Elemento de Cómputo (ARC CE), el recurso obtendrá interfaces del *grid* estándar, a través de las cuales los usuarios, con certificados X.509 validos, pueden obtener información sobre el recurso, enviar, consultar y administrar trabajos con la ayuda de las herramientas del cliente. El recurso también obtendrá la capacidad de registrarse en varios sistemas de información del *grid* de modo que las herramientas del cliente lo descubran.

El cliente de línea de comando de ARC pueden interactuar con A-REX (uno de los *daemons* ejecutándose en ARC CE) u otros elementos de cómputo, admiten varios protocolos de transferencia de datos para poder cargar y descargar archivos de todo tipo de recursos de almacenamiento. Consulta los recursos informáticos disponibles del sistema de

información, haciendo una distribución basada en los requisitos especificados en la descripción del trabajo, puede consultar el estado de los trabajos y administrar su ciclo de vida, y manejar todos los aspectos de la comunicación segura, incluida las credenciales del usuario.

2.1. ARC Computing Element

ARC CE es una interfaz entre el sistema de reparto de trabajos del experimento y un centro de cálculo. El experimento envía el trabajo al CE que lo transforma en el sistema local de envío de trabajos y ficheros. En nuestro caso se ha adaptado para enviar los trabajos, que llegan a nuestra instancia en el PIC, al MareNostrum4.

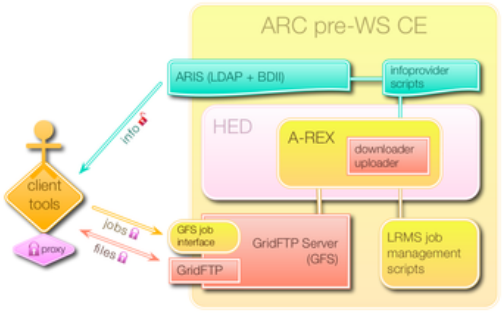

Fig. 1: Estructura interna de ARC CE [10]

Un Elemento de Cómputo tiene las siguiente funciones: [2] (1.3)

- darse a conocer y registrarse en un sistema de información para que las herramientas cliente conozcan su dirección y servicios.
- aceptar las solicitudes de ejecución de trabajos (escritos en lenguajes de descripción de trabajos estándar) a través de la interfaz de envío y procesar los trabajos manejados por el servicio de ejecución.
- aceptar los archivos solicitados por los trabajos del usuario a través de la interfaz de acceso a archivos o descargarlos de almacenamientos remotos y evitar descargar los mismos archivos, almacenándolos en caché múltiples veces.
- reenviar los trabajos al sistema local de gestión de recursos (Slurm, Condor, Torque, OpenPBS, ...) que los programará y ejecutará en los nodos de computación en los recursos locales.
- para supervisar el estado de los trabajos ejecutando los *scripts* del proveedor de información y hacer que esta información esté disponible a través de la interfaz de consulta.
- hacer que los resultados (archivos de salida) de los trabajos sean accesibles a través de la interfaz de acceso o subirlos a un almacenamiento remoto.

El componente más importante del Elemento de Cómputo ARC es el A-REX (ARC Resource-coupled EXecution service). A-REX acepta solicitudes que contienen una descripción de trabajos genéricos y su maneja su ejecución en el sistema de colas local. [2] (1.4)

Se encarga del procesamiento previo y posterior de los trabajos: descarga de archivos que contienen datos de entrada o módulos de programa de una amplia gama de fuentes y almacenamiento o subida de los resultados de salida. ARC CE con la ayuda de A-REX y algunos otros servicios proporciona dos conjuntos distintos de interfaces: las interfaces de servicio pre-web, que se basan en LDAP y GridFTP, y las interfaces de servicio web.

Podemos ver un ejemplo aqui:

nordugrid.org/monitor/atlas/clusdes.php?host=arc-ce01.pic.esport=2135

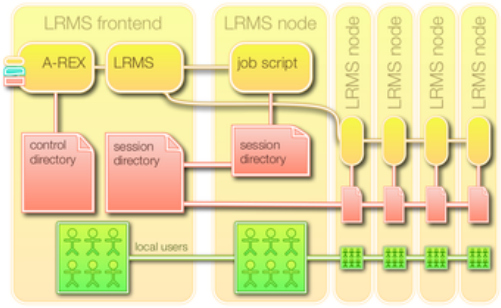

Fig. 2: ARC CE como elemento de entrada a un sistema de colas [10]

Los trabajos en el Elemento de Cómputo ARC generalmente siguen estos pasos: [2] (1.6.1)

1. El cliente se conecta a la interfaz de envío de trabajos (ya sea a la interfaz de servicio web de A-REX o al servidor GridFTP).

   Las herramientas del cliente crean un certificado proxy utilizando las credenciales del usuario y, opcionalmente, recopilan más información sobre la Organización Virtual (VO) a la que pertenece el usuario conectándose a un *Virtual Organization Membership Service* (VOMS).

2. Utilizando los procesos de la infraestructura de clave pública X.509, el cliente y el servidor se autentican entre sí, en función de las credenciales de CA de confianza que se instalaron previamente en ambos extremos.

3. A-REX autoriza al usuario en base a reglas configurables y asigna la identidad del *grid* a un nombre de usuario local que debería estar disponible también en todos los nodos de trabajo.

4. El cliente otorga las credenciales del usuario al A-REX para permitirle actuar en nombre del usuario cuando transfiere archivos.

5. Una descripción del trabajo escrita en uno de los lenguajes admitidos se envía desde el cliente al servidor.

6. Se acepta el trabajo y se crea un directorio (el directorio de sesión, SD) que será el hogar de la sesión. Los metadatos sobre el trabajo se escriben en el directorio de control de A-REX.

7. El cliente recibe la ubicación del directorio de sesión y, si hay archivos de entrada locales, se cargarán en el

SD a través de la interfaz de acceso a archivos (ya sea a través de la interfaz HTTP de A-REX o a través del servidor GridFTP).

8. Si la descripción del trabajo especifica archivos de entrada en ubicaciones remotas, el A-REX obtiene los archivos necesarios y los coloca en el SD. Si el almacenamiento en caché está habilitado, el A-REX comprueba primero si el archivo ya se descargó recientemente y, si es posible, utiliza la versión en caché.

9. Cuando todos los archivos escritos en la descripción del trabajo están presentes, se crea un *script* de trabajo adecuado y se envía al sistema de colas configurado (LRMS).

10. Durante este tiempo, el cliente puede acceder continuamente al SD del trabajo, por lo que se puede verificar cualquier resultado intermedio.

11. Los *scripts* del proveedor de información monitorean periódicamente el estado del trabajo, actualizando la información en el directorio de control.

12. Cuando finaliza el trabajo en el LRMS, A-REX sube, mantiene o elimina los archivos de salida resultantes de acuerdo con la descripción del trabajo.

    El cliente delega el proxy del cliente en el elemento informático, mientras que ambas partes verifican que las credenciales estén firmadas por una autoridad de certificación (CA) de confianza.

13. El cliente también puede descargar los archivos de salida a través de la interfaz de acceso a archivos y eliminar el trabajo del Elemento de Cómputo (CE). Durante toda la vida útil del trabajo, su estado puede consultarse a través de la interfaz LDAP o la interfaz del web.

2.2. VOMS

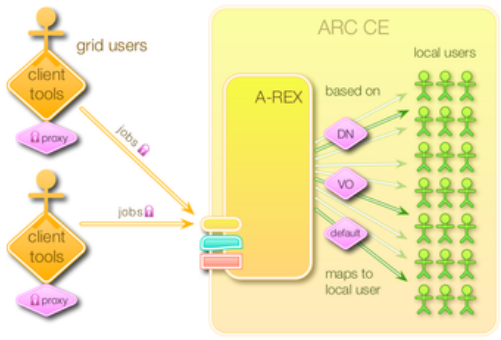

Fig. 3: Exemple de figura [10]

El *Virtual Organization Membership Service* (VOMS) es una autoridad que sirve como repositorio central para la información de autorización de usuario de una VO, brindando soporte para clasificar a los usuarios en jerarquías grupales, realizando un seguimiento de sus roles y otros atributos para emitir certificados y SAML utilizado en el entorno *grid* para fines de autorización. [19]

VOMS se compone de dos componentes principales:

- el servicio central de VOMS, que emite certificados de atributos para clientes autenticados.
- el servicio de administración de VOMS, que es utilizado por el administrador de la VO para administrar las VO y administrar los detalles de sus miembros.

Las asignaciones del *grid* entre los usuarios y las cuentas locales de Unix se enumeran en el archivo `grid-mapfile`, que generalmente se encuentra en el directorio `/etc/grid-security`. Por defecto, este archivo también sirve como una lista de usuarios autorizados. Para facilitar el trabajo del administrador de seguridad, NorduGrid proporciona una colección de *scripts* y trabajos `cron` que mantienen automáticamente sincronizados los archivos de mapa del *grid* local con una base de datos central de usuarios. [2] (4.4.1)

2.3. Puppet

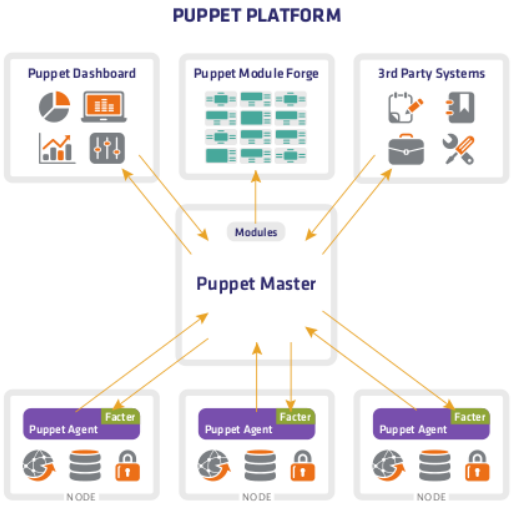

Fig. 4: Esquema de un sistema administrado con puppet [12]

Puppet, un motor administrativo automatizado para sistemas Linux, Unix y Windows, realiza tareas (como agregar usuarios, instalar paquetes y actualizar las configuraciones del servidor) en función de una especificación centralizada. Puppet se configura en una arquitectura maestro-agente, en la que un nodo maestro controla la información de configuración para una flota de nodos de agente gestionados. Puppet Server realiza el rol del nodo maestro. [13]

2.4. Desarrollo

En el comienzo de este trabajo comenzamos con una maquina virtual (arc-ce01.pic.es) que estaba en producción pero se había configurado individualmente, los cambios respecto la maquina base aun se tenian que configurar en los repositorios de del PIC usados para puppet (pic-arcce, r10k_config[arc]).

Lo primero que se hizo fue crear una nueva maquina virtual (arc-ce02.pic.es) donde se probarían los cambios con puppet para comprobar que todo funcionase correctamente. Se configuro la maquina de manera que las carpetas de necesarias para ARC CE estuviesen montadas,

con `sshfs`, remotamente con el nodo de login del MareNostrum4. Asimismo se crearon *scripts* para reemplazar las llamadas a los comandos de Slurm. También se comprobó que el archivo plantilla para generar `arc.conf` hiciese su trabajo correctamente, que las llaves ssh se copiasen en los directorios correctos y que todos los certificados de `/etc/grid-security` tuvieran los permisos correctos.

Después de esto, para que funcionase correctamente, y hacer que el estado del recurso y sus publicaciones fuesen públicos (nordugrid.org/monitor/atlas/...), era necesario añadir permisos de acceso a la red externa y añadir la maquina al recolector de BDII del PIC. En nuestro caso, adicionalmente también se añadió la maquina a los servicios de monitorización del PIC.

Para hacer pruebas se pueden usar las herramientas de linea de comandos del paquete `arcjobtool`, que contiene `arcproxy`, `arctest`, `arcget`, etc..., útiles para identificarse como usuario de ATLAS, enviar y ver resultados de trabajos de pruebas. En nuestro caso se necesita un certificado de ATLAS para establecer un proxy valido.

Una vez que los datos de la maquina son accesibles también hace falta registrar la maquina en GOCDB (goc.egi.eu), por temas de *accounting* (contabilidad de la contribución al proyecto) y añadirla a la cola PanDA en AGIS (atlas-agis.cern.ch/agis/panda-queue/detail/pic_MareNostrum4).

Como la maquina `arc-ce01` tiene que añadirse a puppet y la `arc-ce02` ya esta lista para producción, las intercambiamos mientras reinstalamos y actualizamos la configuración de `arc-ce01`. Primero se pondría la cola en BROKEROFF y esperaríamos a que los trabajos que aun estén en la cola acabasen, entonces, eliminaríamos `arc-ce01` de la cola, añadiriamos `arc-ce02` y cambiaríamos el estatus de la cola a ACTIVE.

Si queremos hacer una prueba final de toda la cadena, después de un tiempo para que la base de datos AGIS se actualice, podemos enviar un trabajo de verdad desde bigpanda (bigpanda.cern.ch)

3 Container

3.1. Singularity

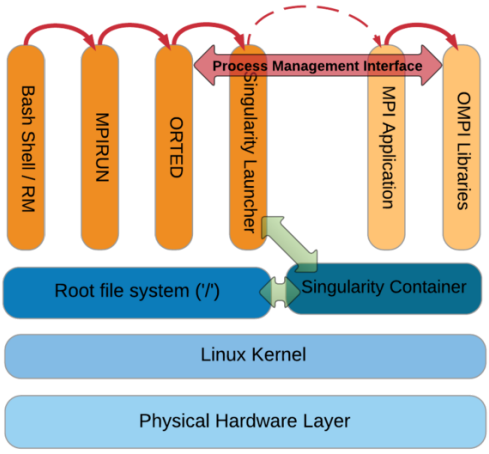

Fig. 5: Diagrama por capas de SW con Singularity [15]

Singularity es una plataforma de contenedores, como Docker, aunque con menos aislamiento. Permite crear y ejecutar contenedores que juntan piezas de software de forma portátil y reproducible. Se puede construir un contenedor utilizando Singularity en una estación de trabajo y luego ejecutarlo en muchos de los clústeres de HPC más grandes del mundo, grupos de universidades o empresas locales, un solo servidor, en la nube... [8]

Singularity se creó para ejecutar aplicaciones complejas en clústeres HPC de una manera simple, portátil y reproducible. Desarrollado por primera vez en el Laboratorio Nacional Lawrence Berkeley, rápidamente se hizo popular en otros sitios de HPC, sitios académicos y más allá. Singularity es un proyecto de código abierto, con una comunidad de desarrolladores y usuarios. La base de usuarios continúa expandiéndose, con Singularity ahora utilizado en toda la industria y académicamente.

Uno de los propósitos más destacables de Singularity es ayudar a hacer ciencia reproducible, los contenedores de singularidad se pueden construir para incluir todos los programas, bibliotecas, datos y *scripts* de manera que se pueda contener una demostración completa y archivarla o distribuirla para que otros la repliquen sin importar la versión de Linux que estén ejecutando actualmente.

3.2. CVFMS

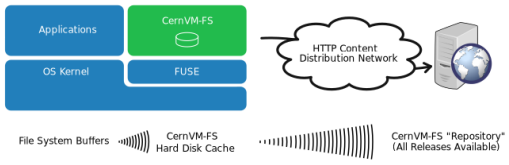

Fig. 6: Diagrama por capas del sistema de ficheros con CVMFS [11]

CernVM-FS proporciona un servicio de distribución de software escalable, confiable y de bajo mantenimiento. Fue desarrollado para ayudar a las colaboraciones de High Energy Physics (HEP) a implementar software en la infraestructura distribuida en todo el mundo utilizada para ejecutar aplicaciones de procesamiento de datos. CVMFS se implementa como un sistema de archivos POSIX de solo lectura en el espacio del usuario (un módulo FUSE). Los archivos y directorios se alojan en servidores web y se montan en el espacio de nombres universal `/cvmfs`. CVMFS utiliza solo conexiones HTTP salientes, por lo que evita la mayoría de los problemas de firewall de otros sistemas de archivos de red. Transfiere datos y metadatos a pedido y verifica la integridad de los datos mediante *hashes* criptográficos. [6]

Mediante el una política de caché agresiva y latencias reducidas, CVMFS se centra específicamente en el caso de uso del software. El software generalmente comprende muchos archivos pequeños que se abren y leen enteros con frecuencia. Además, el caso de uso del software incluye búsquedas frecuentes de archivos en múltiples directorios cuando se examinan las rutas de búsqueda.

CVMFS es utilizado activamente por colaboraciones científicas. En muchos casos, reemplaza al administrador de paquetes y las áreas de software compartido en los sis-

temas de archivos del clúster como medio para distribuir el software utilizado para procesar los datos del experimento.

3.3. Desarrollo

Para ejecutar la simulación se usa una imagen hecho con Singularity que contiene toda la información y programas necesarios. Normalmente estos se leerían de CVMFS, pero desde el MareNostrum4 no hay conexión directa a internet para poder montar el sistema de ficheros.

Ya disponíamos de una imagen, de más de 450GiB, basada en CentOS6 [4] que contenía una copia de `/cvmfs/` [18] y construida con una versión antigua de Singularity con un sistema de ficheros `ext3`, nuestro objetivo es cambiar la imagen base por CentOS7, para conseguirlo lo que es necesario hacer es simplemente copiar `/cvmfs/` a la nueva imagen, respetando los permisos y los propietarios.

Como primer paso podemos extraer `/cvmfs/` y archivarlo en un `tar` con las opciones adecuadas, esta operación podemos esperar que tarde unas 8 horas, el archivo resultante es de un tamaño similar a la imagen. Para continuar necesitamos alguna forma de añadirlo a una imagen de Singularity, se puede intentar des-archivar desde dentro del contenedor, pero las versiones nuevas de Singularity crean imágenes de solo lectura en formato SIF y sin posibilidad de añadir espacio extra a la imagen, así que fallará por falta de espacio.

Esto nos deja con dos opciones: crear la imagen en formato *sandbox* o a partir de un *recipe file*. En ambos casos necesitamos extraer el archivo tar, lo que supone un problema porque la maquina virtual tiene un espacio de disco local más limitado que lo que ocupa `/cvmfs/`, y las opciones `--same-owner` y `--same-permissions` necesitan permisos de root para hacer `chown`, pero el almacenamiento que tiene suficiente capacidad esta montado como NFS sin `no_root_squash`.

Si no tenemos permisos de root en el servidor de NFS, lo que podemos hacer es crear un archivo de 500GiB y usarlo como un *block device*. Usamos `mkfs` para crear un sistema de fichero donde tenemos permitido hacer `chown` y tiene la capacidad necesaria para des-archivar. Esta operación puede tardar unos 5 días, pero, una vez extraído, podemos extraer la imagen de CentOS7, que podemos obtener de `/cvmfs/atlas.cern.ch/repo/containers/images/singularity/x86_64-centos7.img` en un directorio *sandbox*, mover la carpeta a la raíz de la imagen y construirla.

El único problema que nos queda por resolver es que la versión que se usa en el MareNostrum4, que es la misma que tenemos instalada, tiene un *bug* [3] por el cual no conserva los propietarios de los archivos que copia a dentro de la imagen.

Por suerte, el día en el que me encontré con el problema ya se había hecho un *pull request* con una solución [16] y, en breve, salio una versión nueva que sirvió para crear la imagen correctamente y era compatible con la versión en el MareNostrum4. [14]

Lo ultimo es hacer una nueva versión de la imagen aplicando el parche [9] para evitar redundancia de los eventos al ejecutar con MPI usando Harvester.

4 Resultados

Los resultados siguientes se comprenden entre 2019/11/01 y 2020/01/17, se han obtenido a partir de los servicio de monitorización del CERN. [7]

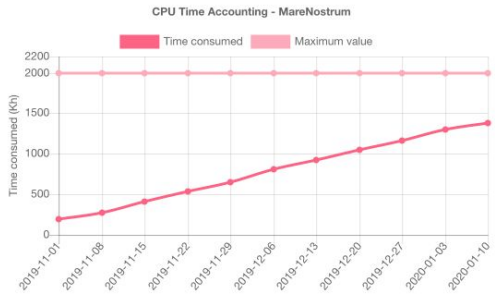

Fig. 7: Tiempo consumido según MareNostrum4

Fig. 8: Gráfico acumulativo de los trabajos completados en la cola pic_MareNostrum

Fig. 9: Recursos usados en la cola pic_MareNostrum

En las figuras 7 y 8 podemos ver como ha ido que se han ido consumiendo el tiempo de cómputo asignado al PIC en el MareNostrum4. También vemos en la figura 9 los trabajos se han estado ejecutando de manera estable, el primer pico corresponde a una prueba de estrés, en el resto del gráfico se puesto un limite a la cola para que las 2 millones de horas concedidas al PIC de cómputo en el MareNostrum4 duren durante el periodo establecido.

Podemos ver que el desempeño, respecto al porcentaje de errores, de los trabajos es ligeramente peor que el vemos en el PIC (fig. 10, 12), aunque es igual que centros como el IFIC. Esto se debe en mayor parte al hecho de que el PIC es un centro de datos *Tier 1* de ATLAS que contiene los datos requeridos para la simulación, en el caso del MN4 podemos ver en la figura 11 que el error más común es `DDM Error` que se produce durante la transferencia de datos. Se puede

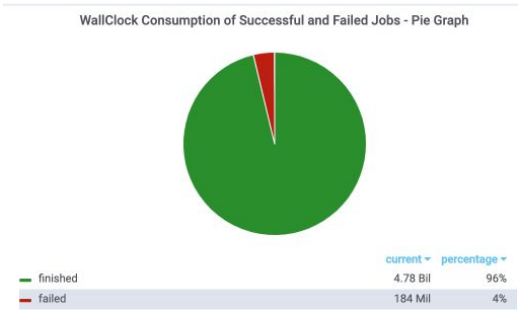

Fig. 10: Porcentaje de errores de los trabajos enviados al MN4

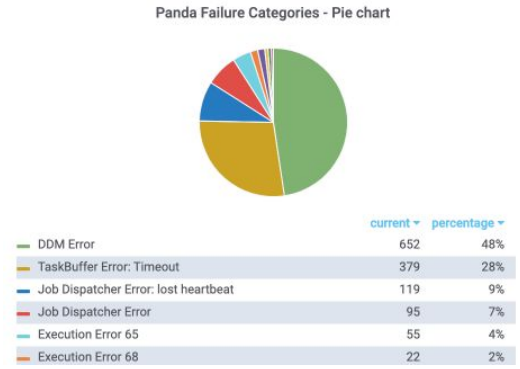

Fig. 11: Porcentaje de errores por tipo de los trabajos enviados al MN4

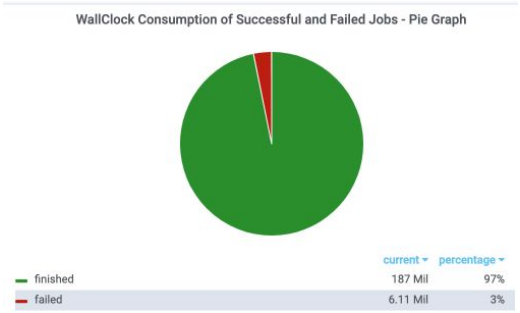

Fig. 12: Porcentaje de errores de los trabajos ejecutados en el PIC

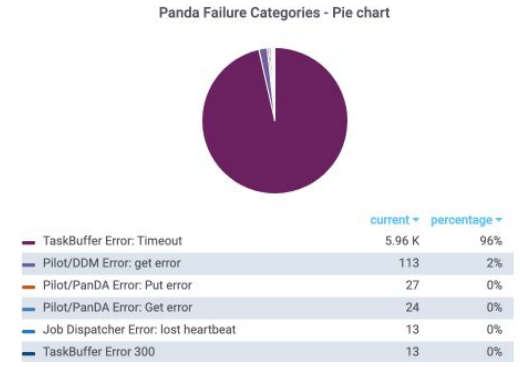

Fig. 13: Porcentaje de errores por tipo de los trabajos ejecutados en el PIC

observar que la gran mayoría de los restantes son debidos al gestor de colas en diferentes puntos del envio de trabajos tanto en fig. 11 como en fig. 13.

5 Conclusiones

Aunque no hemos tenido tiempo de acabar todo lo que nos habíamos propuesto en un principio, si que hemos acabado con un sistema funcional que se ha puesto en producción y ha estado funcionando meses sin problemas.

A pesar de que si hemos probado a ejecutar alguna imagen *single release* de ATLAS, no hemos hecho los cambios necesarios en la cola de trabajos para que las simulaciones se ejecuten con estas automáticamente.

También ha quedado por cumplir en objetivo 5, aún se puede mejorar la ejecución con MPI e intentar hacer que el programa eventualmente funcione sin la necesidad de Singularity.

6 Trabajos futuros

Para empezar se debería acabar el objetivo 5, y, como sabemos que las imágenes *single release* de ATLAS funcionan, ponerlas en producción parece el siguiente paso a seguir.

Durante el desarrollo del proyecto también se discutió de que las maquinas ARC CE podrían hacer funcionar con Kubernetes.

Por ultimo, la financiación del MareNostrum5 ha sido aprobada su en Junio del 2019 por la Unión Europea y estará basado en procesadores ARM. Un objetivo interesante usar una instancia existente en el BSC y hacer pruebas de uso.

Agradecimientos

En primer lugar me gustaría agradecerle la tutorización por parte de Andreu Pacheco Pages del PIC-IFAE por instruirme y guiarme durante el desarrollo de este trabajo; y a Andreu Pérez Martínez por parte de la UAB.

También me gustaría agradecerle a Carles Acosta por ponerme al día con la infraestructura ya funcional de ARC a lo largo del proyecto, y al resto del equipo del PIC por la ayuda con la creación de instancias de maquinas virtuales y resolución de alguna que otra duda técnica.

Referencias

Información

[1] ATLAS Collaboration. *Computing Activities at the Spanish Tier-1 and Tier-2s for the ATLAS experiment towards the LHC Run3 and High Luminosity (HL-LHC) periods.* 2020 [unpublished]. (accessed: 2020.01.17).

[2] *ARC Computing Element: System Administrator Guide*. 2019. URL: `http://www.nordugrid.org/documents/arc-ce-sysadm-guide.pdf`. (accessed: 2020.01.17).

[3] *All files in container owned by root · Issue 2860*. URL: `https://github.com/sylabs/singularity/issues/2860`. (accessed: 2020.01.17).

[4] *atlas-fat-container: Build ATLAS fat singularity and shifter container images via uncvmfs*. URL: https://github.com/wyang007/atlas-fat-container. (accessed: 2020.01.17).

[5] *atlas/athena Tags - Docker Hub*. URL: https://hub.docker.com/r/atlas/athena/tags. (accessed: 2020.01.17).

[6] *CernVM File System*. URL: https://cernvm.cern.ch/portal/filesystem. (accessed: 2020.01.17).

[8] *Introduction to Singularity — Singularity container 3.5 documentation*. URL: https://sylabs.io/guides/3.5/user-guide/introduction.html. (accessed: 2020.01.17).

[13] *puppet: Server automation framework and application*. URL: https://github.com/puppetlabs/puppet. (accessed: 2020.01.17).

[14] *Singularity 3.5.1 Release*. URL: https://github.com/sylabs/singularity/releases/tag/v3.5.1. (accessed: 2020.01.17).

[16] *Small Hotfix for Issue 2860*. URL: https://github.com/sylabs/singularity/pull/4796. (accessed: 2020.01.17).

[17] *Spanish ATLAS Tier-1 and Tier-2 perspective on computing over the next years*. URL: https://www.researchgate.net/publication/335862680_Spanish_ATLAS_Tier-1_Tier-2_perspective_on_computing_over_the_next_years. (accessed: 2020.01.17).

[18] *uncvmfs*. URL: https://github.com/ic-hep/uncvmfs. (accessed: 2020.01.17).

[19] *VOMS Service — IT Department*. URL: http://information-technology.web.cern.ch/services/VOMS-service. (accessed: 2020.01.17).

REQUIEREN CUENTA DEL CERN

[7] *Grafana (CERN)*. URL: https://monit-grafana.cern.ch/d/000000696/job-accounting-historical-data?orgId=17. (accessed: 2020.01.17).

[9] *JIRA (CERN)*. URL: https://its.cern.ch/jira/browse/ADCINFR-102. (accessed: 2020.01.17).

IMÁGENES

[10] *NorduGrid — ARC Compute Element*. URL: http://www.nordugrid.org/arc/ce/. (accessed: 2020.01.17).

[11] *Overview — CernVM-FS 2.8.0 documentation*. URL: https://cvmfs.readthedocs.io/en/latest/cpt-overview.html. (accessed: 2020.01.17).

[12] *Puppet Labs Announces Puppet Enterprise*. URL: http://www.linux-magazine.com/Online/News/Puppet-Labs-Announces-Puppet-Enterprise. (accessed: 2020.01.17).

[15] *Singularity: A Container for HPC*. URL: http://www.admin-magazine.com/HPC/Articles/Singularity-A-Container-for-HPC. (accessed: 2020.01.17).